Lista de Exercícios - Alocação Dinâmica

1. a) Escreva uma função que recebe como parâmetro dois valores inteiros `tam` e `lim`, aloca dinamicamente um vetor de tamanho `tam`, preenche esse vetor com valores aleatórios de `0` a `lim` e retorna esse vetor alocado e preenchido.

   Protótipo: `int * criaVetor(int tam, int lim);`

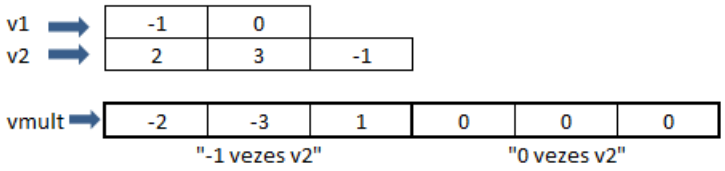
   b) Escreva uma função que inclua `n` novos elementos num vetor previamente alocado e preenchido como no item a).

   Protótipo: `int * expandeVetor(int *v, int tam, int n, int lim);`

   Desafio: Modifique a função do item a) para que seja do tipo `void`. Para isso vc deverá considerar que a função recebe o endereço de um ponteiro (**) (que estava apontando para lugar desconhecido e que agora deverá apontar para o bloco alocado). A função deve preencher o vetor com valores aleatórios de `-lim` a `lim`.

   Protótipo: `void criaVetorPorReferencia(int **v, int tam, int lim);`

2. Escreva um programa que aloque memória para dois vetores, com tamanhos diferentes, dados pelo usuário. Os vetores devem ser preenchidos com valores aleatórios no intervalo `[0, 10]`. O programa deve então criar um terceiro vetor, capaz de conter todos os elementos `Vi*Vj`, onde `Vi` é um elemento do primeiro vetor e `Vj` é um elemento do segundo vetor (veja um exemplo na figura abaixo). A multiplicação elemento-a-elemento dos dois vetores deve ser calculada, e os valores obtidos devem ser apresentados. Dica: você pode usar uma ou mais funções para organizar o seu código.



3. Escreva uma função que recebe como parâmetro um vetor de int. A função deve criar outro vetor, e colocar nele os elementos do vetor passado como parâmetro, mas sem repetições. Por exemplo, se os elementos do vetor original são `[0, 1, 2, 3, 4, 3, 2, 4, 5, 3, 2, 6, 1, 0]`, o novo vetor deve conter apenas `[0, 1, 2, 3, 4, 5, 6]`. O novo vetor deve ter apenas o tamanho necessário para manter os valores. Os dados do vetor original devem ser mantidos intactos. No final, a função deve retornar o número de itens do vetor criado - o vetor em si é retornado como um parâmetro passado por referência.

4. (a) Construa uma função, denominada `custo_cidades` que, dado um vetor de inteiros com o código das cidades na ordem que deverão ser visitadas, o número de cidades visitadas no percurso e a matriz de custos, retorne o custo total do itinerário. A função deve ter o seguinte protótipo:

   `int custo_cidades (int* cidades, int n_cidades, int** m);`

   (b) a função `main`, que deve

   - solicitar ao usuário a dimensão da matriz de custo,
   - alocar memória para esta matriz utilizando uma função denominada `alocaMatriz`,
   - preencher a matriz com dados digitados pelo usuário,
   - solicitar ao usuário o tamanho do itinerário,
   - alocar memória para o vetor que irá armazenar o itinerário usando uma função denominada `alocaVetor`,
   - preencher o vetor de itinerários
   - chamar a função `custo_cidades` e imprimir na tela o custo.

   Dica: Vale a pena observar que a matriz `m` sempre será quadrada ($n \times n$); e o custo de transporte entre as cidades i e j é dado pelo elemento m_{ij} da matriz. Exemplo: Considerando a seguinte matriz de custos

$$m_{4\times4} = \begin{pmatrix} 0 & 31 & 25 & 1 \\ 23 & 0 & 1 & 8 \\ 75 & 1 & 0 & 400 \\ 1 & 15 & 20 & 0 \end{pmatrix}$$

   O custo do itinerário $\{0, 3, 1, 2, 1, 0\}$ é calculado da seguinte forma: $m_{03}+m_{31}+m_{12}+m_{21}+m_{10} = 1+15+1+1+23 = 41$

Extra: Listas de Vetores e Matrizes agora com alocação Dinâmica

5. Refaça alguns (ou todos) os exercícios da lista de vetores usando alocação dinâmica.
6. Refaça alguns (ou todos) os exercícios da lista de matrizes usando alocação dinâmica.